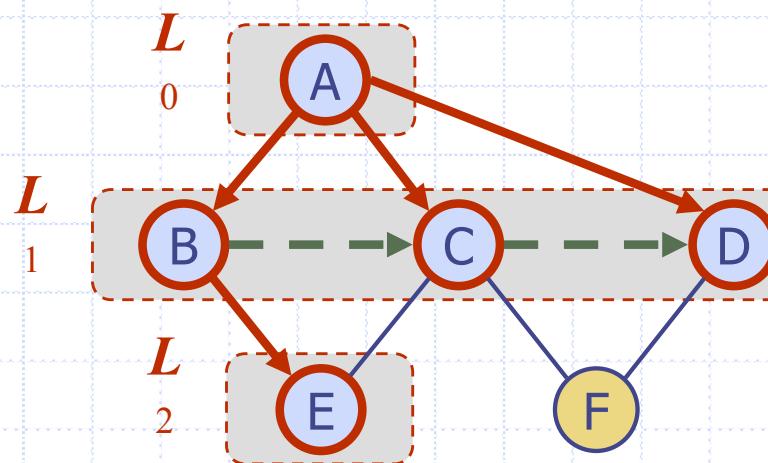


Presentación para usar con el libro de texto **Algorithm Design and Applications**, de MT Goodrich y R. Tamassia, Wiley, 2015

Búsqueda en Amplitud



Búsqueda en Amplitud

- La búsqueda en amplitud (BFS) es una técnica general para recorrer un grafo
- Un recorrido BFS de un grafo G
 - Visita todos los vértices y aristas de G
 - Determina si G está conectado
 - Calcula las componentes conexas de G
 - Calcula un bosque de expansión de G
- BFS en un grafo con n vértices y m aristas toma $O(n + m)$ tiempo
- BFS se puede ampliar aún más para resolver otros problemas de grafos
 - Encontrar un camino con el número mínimo de aristas entre dos vértices dados
 - Encuentre un ciclo simple, si lo hay.

Algoritmo BFS

- El algoritmo utiliza “niveles” L_i y un mecanismo para establecer y obtener “etiquetas” de vértices y aristas.

Algorithm $\text{BFS}(G, s)$:

Input: A graph G and a vertex s of G

Output: A labeling of the edges in the connected component of s as discovery edges and cross edges

Create an empty list, L_0

Mark s as explored and insert s into L_0

$i \leftarrow 0$

while L_i is not empty **do**

 create an empty list, L_{i+1}

for each vertex, v , in L_i **do**

for each edge, $e = (v, w)$, incident on v in G **do**

if edge e is unexplored **then**

if vertex w is unexplored **then**

 Label e as a discovery edge

 Mark w as explored and insert w into L_{i+1}

else

 Label e as a cross edge

$i \leftarrow i + 1$

Algoritmo BFS

```
def barrido_amplitud(grafo, vertice):
    """Barrido en amplitud del grafo."""
    cola = Cola()
    while(vertice is not None):
        if(not vertice.visitado):
            vertice.visitado = True
            arribo(cola, vertice)
            while(not cola_vacia(cola)):
                nodo = atencion(cola)
                print(nodo.info)
                adyacentes = nodo.adyacentes.inicio
                while(adyacentes is not None):
                    adyacente = buscar_vertice(grafo, adyacentes.destino)
                    if(not adyacente.visitado):
                        adyacente.visitado = True
                        arribo(cola, adyacente)
                    adyacentes = adyacentes.sig
                vertice = vertice.sig
```

Ejemplo



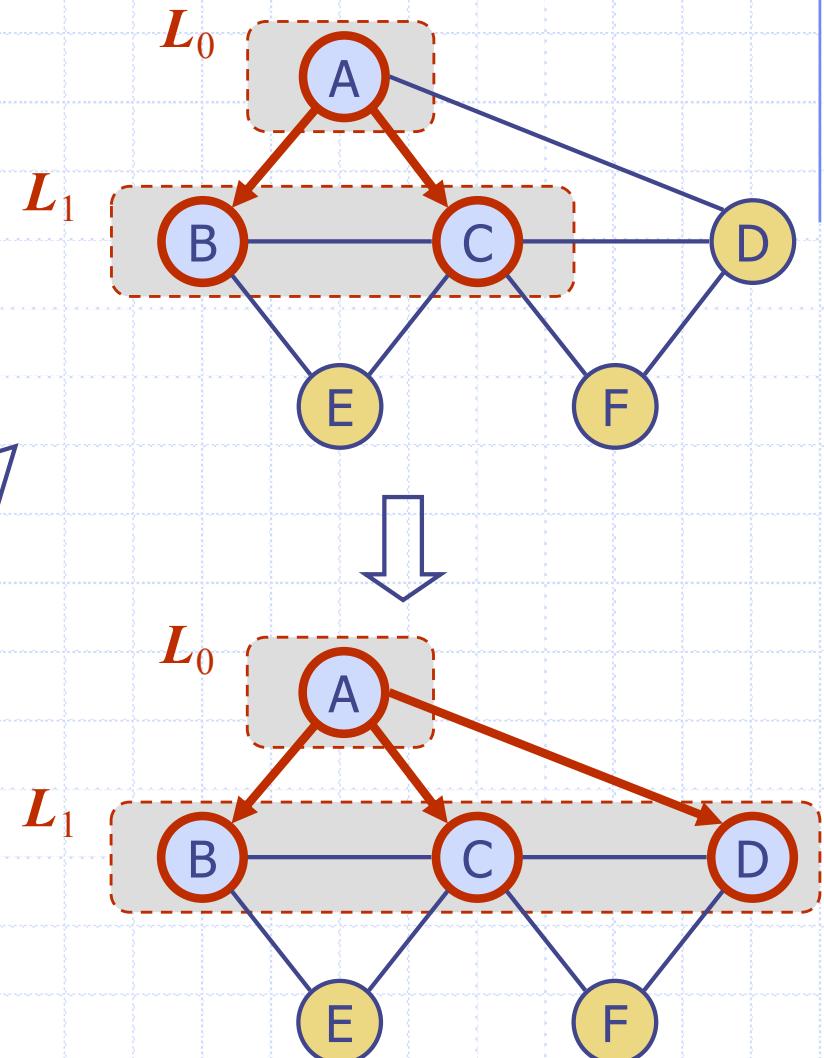
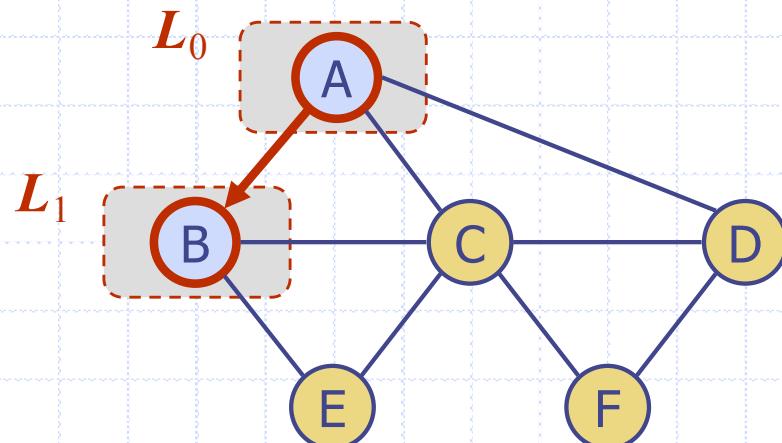
vértice inexplorado

vértice visitado

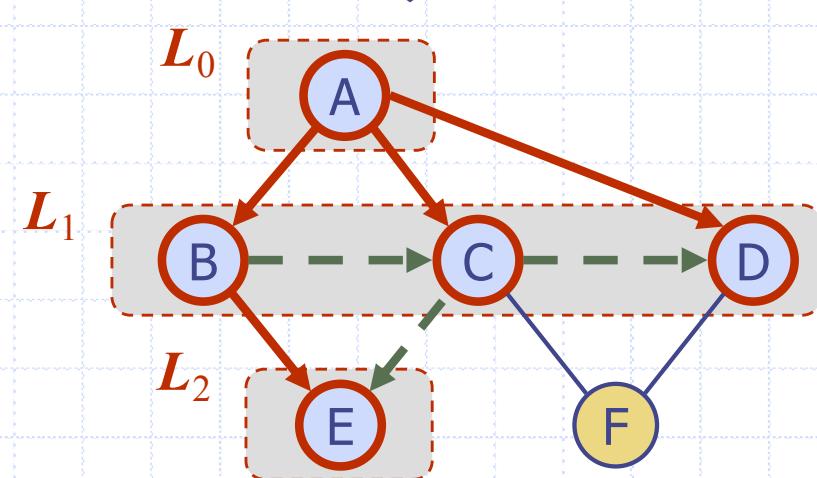
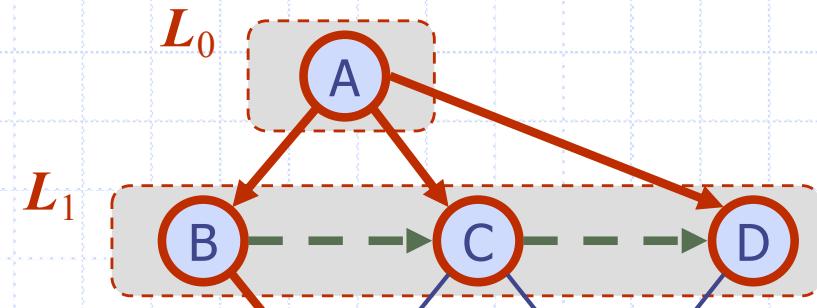
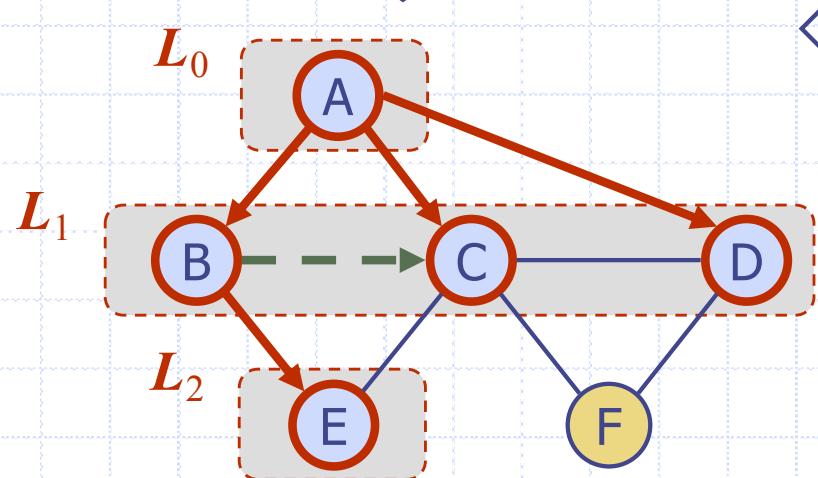
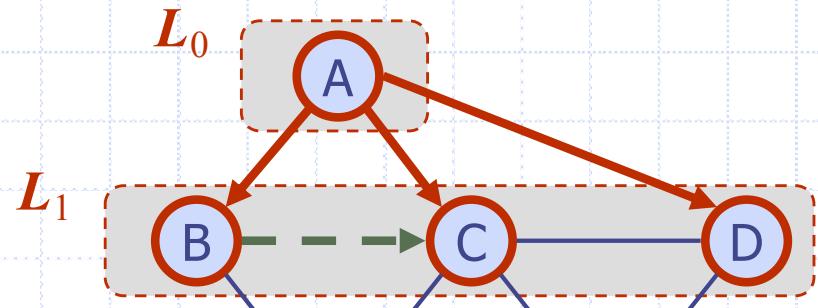
arista inexplorada

arista visitada

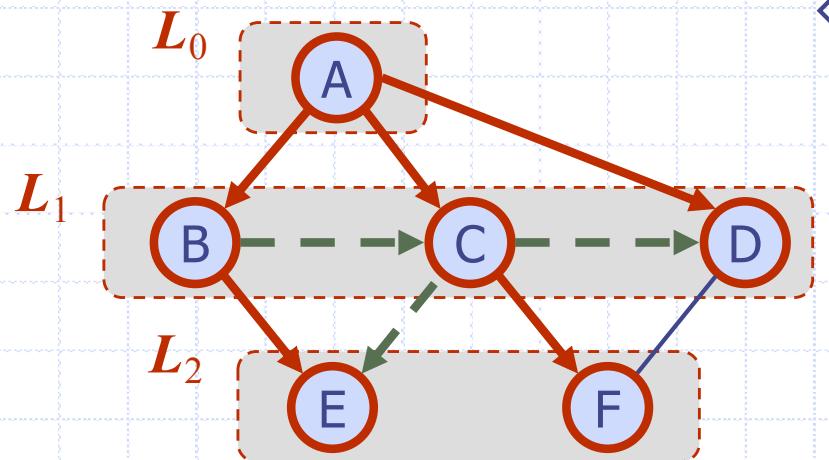
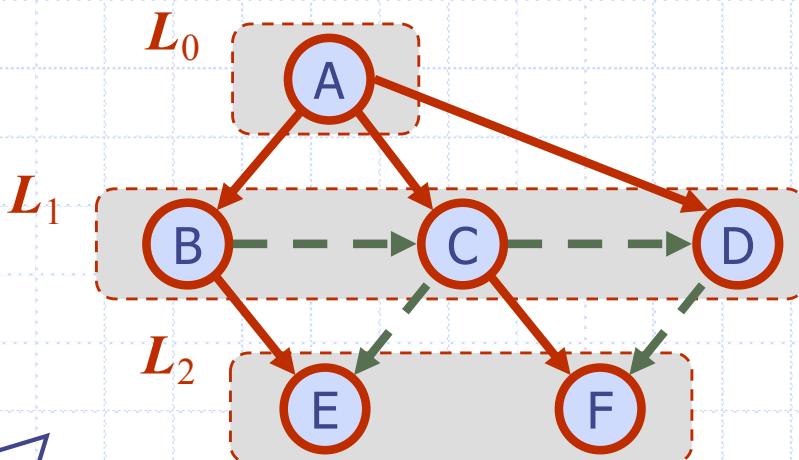
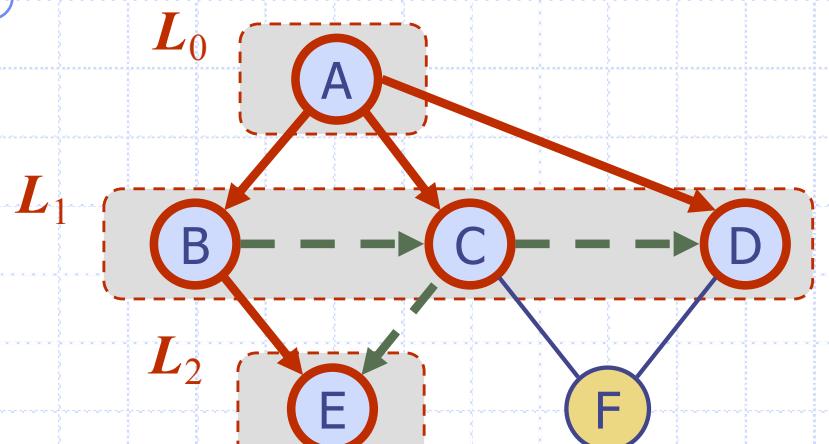
arista cruzada



Ejemplo (continuación)



Ejemplo (continuación)



Propiedades

Notación

G_s : componente conexa de s

Propiedad 1

$BFS(G, s)$ visita todos los vértices y aristas de G_s

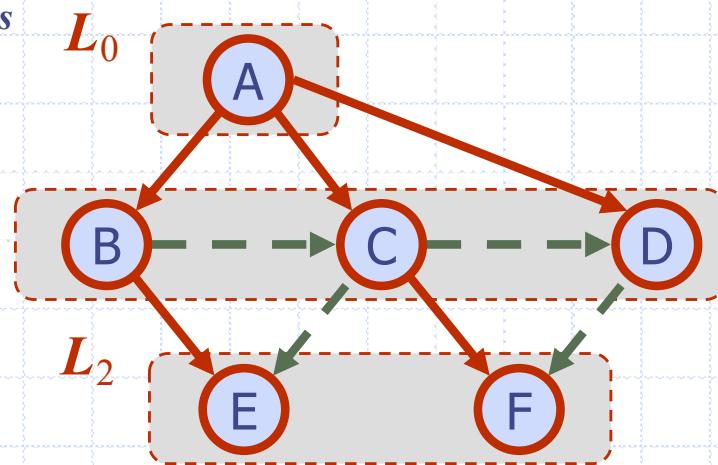
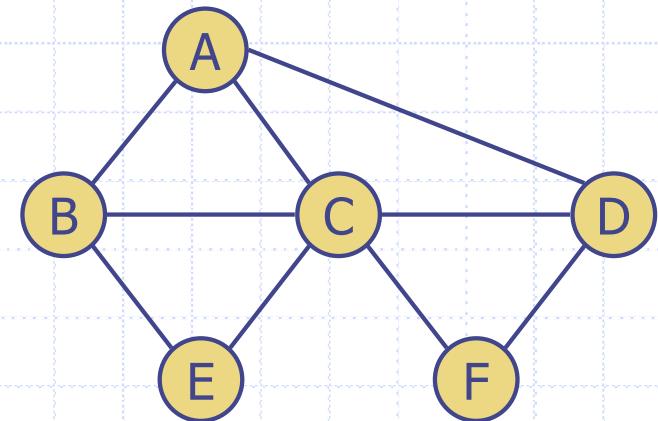
Propiedad 2

Los aristas etiquetadas por $BFS(G, s)$ forman un árbol de expansión T_s de G_s

Propiedad 3

Para cada vértice v en L_i

- El camino de T_s de s a v tiene i aristas
- Cada camino de s a v en G_s tiene al menos i bordes



Análisis

- Establecer/obtener una etiqueta de vértice/borde lleva $O(1)$ tiempo
- Cada vértice está etiquetado dos veces
 - una vez como INEXPLORADO
 - una vez como VISITADO
- Cada arista está etiquetada dos veces
 - una vez como INEXPLORADO
 - una vez como VISITADO o CRUZE
- Cada vértice se inserta una vez en una secuencia L_i
- El método **aristasIncidentes** se llama una vez para cada vértice
- BFS se ejecuta en tiempo $O(n + m)$ siempre que el grafo esté representado por la estructura de lista de adyacencia
 - Recuerde que $\sum_v \deg(v) = 2m$

Aplicaciones

- Podemos usar el algoritmo de recorrido BFS, para un grafo G , para resolver los siguientes problemas en tiempo $O(n + m)$
 - Calcular las componentes conexas de G
 - Calcular un bosque de expansión de G
 - Encontrar un ciclo simple en G , o informar si G es un bosque
 - Dados dos vértices de G , encontrar un camino en G con el mínimo número de aristas, o que no existe tal camino

DFS vs. BFS

Aplicaciones

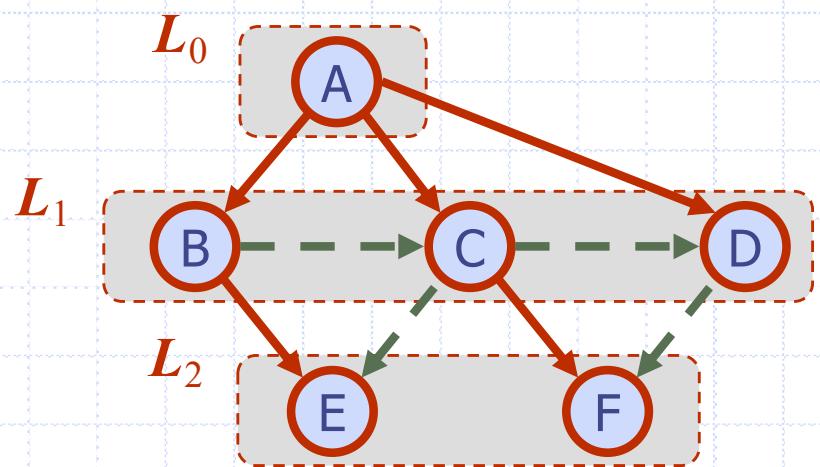
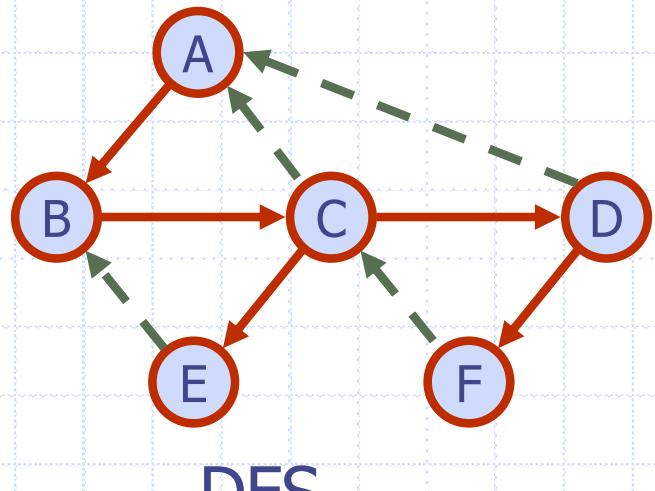
Bosque de expansión, componentes conectados, caminos, ciclos

Caminos más cortos

Componentes biconectados

SFD

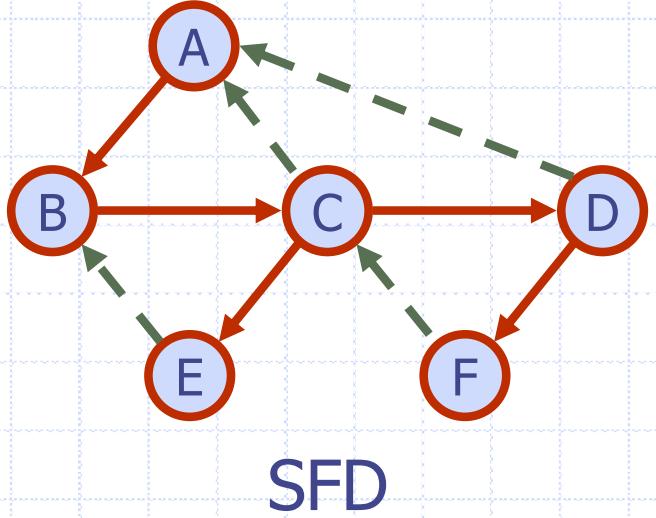
BFS



DFS vs. BFS (Cont.)

Arista de retorno (v, w)

- w es un antepasado de v en el árbol de aristas visitadas



Arista de cruce (v, w)

- w está en el mismo nivel que v o en el siguiente nivel

